Statistical Natural Language Processing

Machine learning: evaluation

Çağrı Çöltekin

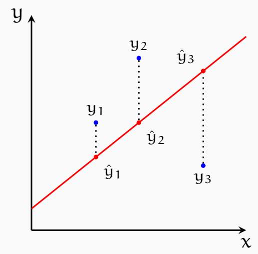
University of Tübingen
Seminar für Sprachwissenschaft

Summer Semester 2018

Measuring success/failure in regression

Root mean squared error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2}$$

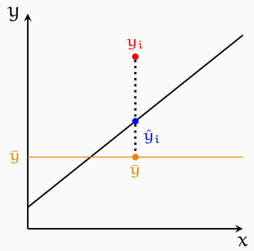


- Measures average error in the units compatible with the outcome variable

Measuring success/failure in regression

Coefficient of determination

$$R^2 = \frac{\sum_i^n(\hat{y}_i - \bar{y})^2}{\sum_i^n(y_i - \bar{y})^2} = 1 - \frac{\text{MSE}}{\sigma_y^2}$$



- r^2 is a standardized measure in range $[0, 1]$
- Indicates the ratio of variance of y explained by x
- For single predictor it is the square of the correlation coefficient r

Measuring success in classification

Accuracy

- In classification, we do not care (much) about the average of the error function
- We are interested in how many of our predictions are correct
- Accuracy measures this directly

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}$$

Accuracy may go wrong

- Think about a 'dummy' search engine that always returns an empty document set (no results found)
- If we have
  - 1 000 000 documents
  - 1000 relevant documents (including the term in the query)

  the accuracy is:

$$\frac{999\,000}{1\,000\,000} = 99.90\,\%$$

- In general, if our class distribution is *skewed* accuracy will be a bad indicator of success

Measuring success in classification

Precision, recall, F-score

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1\text{-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

		true value	
		positive	negative
predicted	pos.	TP	FP
	neg.	FN	TN

Example: back to the search engine

- We had a 'dummy' search engine that returned false for all queries
- For a query
  - 1 000 000 documents
  - 1000 relevant documents

$$\text{accuracy} = \frac{999\,000}{1\,000\,000} = 99.90\,\%$$

$$\text{precision} = \frac{0}{1\,000\,000} = 0\,\%$$

$$\text{recall} = \frac{0}{1\,000\,000} = 0\,\%$$

Precision and recall are asymmetric,
the choice of the 'positive' class is important.

Classifier evaluation: another example

Consider the following two classifiers:

		true value		true value	
		positive	negative	positive	negative
predicted	pos.	7	9	1	3
	neg.	3	1	9	7

Accuracy both $8/20 = 0.4$
Precision $7/16 = 0.44$ and $1/4 = 0.25$
Recall $7/10 = 0.7$ and $1/10 = 0.1$
F-score 0.54 and 0.14

Multi-class evaluation

- For multi-class problems, it is common to report average precision/recall/f-score
- For C classes, averaging can be done two ways:

$$\text{precision}_M = \frac{\sum_i^C \frac{TP_i}{TP_i+FP_i}}{C} \qquad \text{recall}_M = \frac{\sum_i^C \frac{TP_i}{TP_i+FN_i}}{C}$$

$$\text{precision}_\mu = \frac{\sum_i^C TP_i}{\sum_i^C TP_i + FP_i} \qquad \text{recall}_\mu = \frac{\sum_i^C TP_i}{\sum_i^C TP_i + FN_i}$$

(M = macro, μ = micro)
- The averaging can also be useful for binary classification, if there is no natural positive class

Confusion matrix

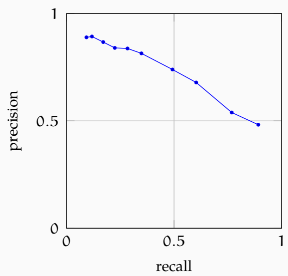
- A confusion matrix is often useful for multi-class classification tasks

		true class		
		a	b	c
predicted	a	10	3	4
	b	2	12	8
	c	0	7	7

- Are the classes balanced?
- What is the accuracy?
- What is per-class, and averaged precision/recall?

Precision–recall trade-off

- Increasing precision (e.g., by changing a hyperparameter) results in decreasing recall
- Precision–recall graphs are useful for picking the correct models
- *Area under the curve* (AUC) is another indication of success of a classifier



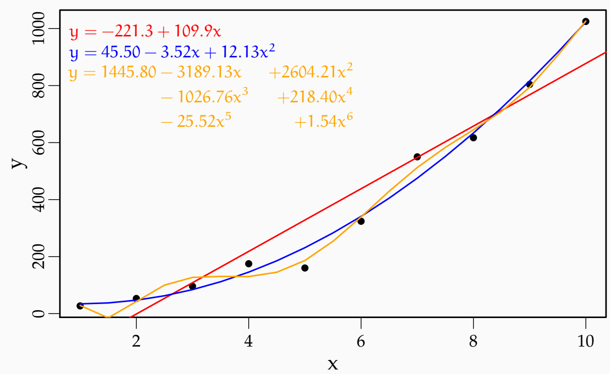
Performance metrics a summary

- Accuracy does not reflect the classifier performance when class distribution is skewed
- Precision and recall are binary and asymmetric
- For multi-class problems, calculating accuracy is straightforward, but others measures need averaging
- These are just the most common measures: there are more
- You should understand what these metrics measure, and use/report the metric that is useful for the purpose

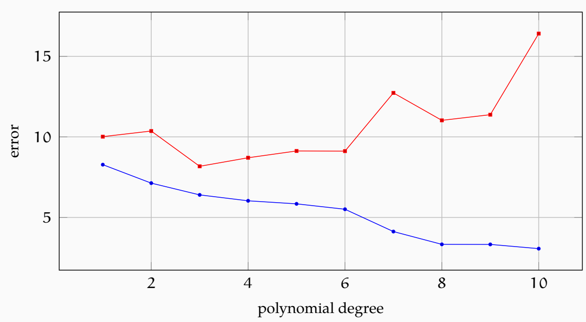
Model selection/evaluation

- Our aim is to fit models that are (also) useful outside the training data
- Evaluating a model on the training data is wrong: complex models tend to fit to the noise in the training data
- The results should always be tested on a test set that does not overlap with the training data
- Test set is ideally used only once - to evaluate the final model
- Often, we also need to tune the model, find best *hyperparameters* (e.g., regularization constant)
- Tuning has to be done on a separate development set

Back to polynomial regression



Training/test error



Bias and variance (revisited)

Bias of an estimate is the difference between the value being estimated, and the expected value of the estimate

$$B(\hat{w}) = E[\hat{w}] - w$$

- An *unbiased* estimator has 0 bias

Variance of an estimate is, simply its variance, the value of the squared deviations from the mean estimate

$$var(\hat{w}) = E\left[(\hat{w} - E[\hat{w}])^2\right]$$

w is the parameters that define the model

Bias–variance relationship is a trade-off: models with low bias result in high variance.

Some issues with bias and variance

- *Overfitting* occurs when the model learns the idiosyncrasies of the training data
- *Underfitting* occurs when the model is not flexible enough for the data at hand
- Complex models tend to overfit – and exhibit high variance
- Simple models tend to show low variance, but likely to have (high) bias

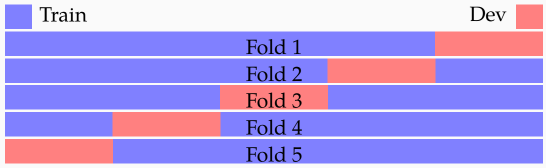
Model selection & hyperparamater tuning

- Our aim is to reduce the test error
- We can estimate the test error on a *development set*, or *held-out* data:
  - Split the data at hand as *training* and *development* set
  - Train alternative models (different hyperparameters) on the training set
  - Choose the model with best development set performance

Cross validation

- To avoid overfitting, we want to tune our models on a *development set*
- But (labeled) data is valuable
- Cross validation is a technique that uses all the data, for both training and tuning with some additional effort
- Besides tuning hyper-parameters, we may also want to get 'average' parameter estimates over multiple folds

K-fold Cross validation



- At each fold, we hold part of the data for testing, train the model with the remaining data
- Typical values for k is 5 and 10
- In *stratified* cross validation each fold contains (approximately) the same proportions of class labels.
- A special case, when k is equal to n (the number of data points is called *leave-one-out cross validation*

The choice of k in k-fold CV

- Increasing k
  - reduces the bias: the estimates converge to true value of the measure (e.g., accuracy) in the limit
  - increases the variance: smaller held-out sets produce more varied parameter estimates
  - is generally computationally expensive
- 5- or 10-fold cross validation is common practice (and found to have a good balance between bias and variance)

Comparing with a basline

- The performance measures are only meaningfull if we have something to compare against

random: does the model do anything useful at all?
majority class: does the classifier better than predicting the majority class all the time?
state-of-the-art: how does your model compare against known (non-trivial) models?

- Diferences between models are reliable only if the same data set is used
- Differences are stable if your test set size is large enough
- Use statistical tests when comparing different models/methods

Summary

The first principle is that you must not fool yourself and you are the easiest person to fool. – Richard P. Feynman

- The measures of success in ML systems include
  - RMSE / r^2
  - Accuracy
  - Precision / recall / F-score
- We want models with low bias and low variance
- Evaluating ML system requires special care:
  - Never use your test set during training / development
  - Tuning your system on a development set
  - Cross-validation allows efficient use of labeled data

Next:

- Have good holiday! We'll start with sequence learning after the break.